PENERAPAN MODEL VIEW CONTROLLER UNTUK MENGHITUNG RATA-RATA NILAI MAHASISWA

Firgo Septiansyah[1], Yuniana Cahyaningrum[2]

Pendidikan Teknologi Informasi

IKIP PGRI Bojonegoro

Email : firgoseptians129@gmail.com [1], yuniana@ikippgribojonegoro.ac.id [2]

Abstrak : Artikel ini membahas penerapan Model View Controller (MVC) dalam menghitung rata-rata nilai mahasiswa. MVC adalah pendekatan desain yang terdiri dari tiga komponen utama: Model, View, dan Controller. Dalam konteks ini, Model digunakan untuk mengelola data nilai mahasiswa, View bertanggung jawab untuk menampilkan informasi kepada pengguna, dan Controller berfungsi sebagai penghubung antara Model dan View. Dalam artikel ini, kami menjelaskan langkah-langkah implementasi MVC dalam penghitungan rata-rata nilai mahasiswa. Pertama, data nilai mahasiswa dikumpulkan dan disimpan dalam Model. Model ini akan mengelola operasi perhitungan nilai, termasuk menjumlahkan nilai dan menghitung rata-ratanya Kemudian, View digunakan untuk menampilkan hasil perhitungan kepada pengguna. Pengguna dapat melihat rata-rata nilai mahasiswa secara jelas dan mudah dipahami melalui tampilan yang disediakan oleh View. Selain itu, Controller bertanggung jawab untuk menerima input pengguna dan menghubungkannya dengan Model yang sesuai. Controller akan menerima perintah dari pengguna untuk menghitung rata-rata nilai mahasiswa dan mengaktifkan operasi perhitungan yang diperlukan di dalam Model. Setelah hasil perhitungan tersedia, Controller akan memperbarui View untuk menampilkan rata-rata nilai yang dihasilkan. Dengan menerapkan pendekatan MVC ini, pengembangan aplikasi penghitungan rata-rata nilai mahasiswa menjadi lebih terstruktur dan modular. Setiap komponen (Model, View, dan Controller) memiliki tanggung jawab yang jelas dan dapat diubah atau dikembangkan secara terpisah tanpa mempengaruhi komponen lainnya. Dalam kesimpulan, artikel ini menggambarkan penerapan Model View Controller dalam konteks penghitungan rata-rata nilai mahasiswa. Pendekatan MVC ini dapat meningkatkan struktur dan keterbacaan kode, memisahkan logika perhitungan dari tampilan, dan memungkinkan pengembangan yang lebih mudah dan fleksibel.

Kata kunci : *Model, View, Controller, rata-rata mahasiswa,*

Pendahuluan

Pengelolaan data dan perhitungan nilai mahasiswa merupakan salah satu aspek penting dalam dunia pendidikan. Dalam era digital yang semakin berkembang, penggunaan teknologi untuk mempermudah dan meningkatkan efisiensi proses tersebut sangatlah relevan. Salah satu pendekatan desain yang dapat diterapkan dalam pengembangan aplikasi penghitungan nilai mahasiswa adalah Model View Controller (MVC).

Model View Controller (MVC) adalah sebuah paradigma desain yang memisahkan logika bisnis (Model), tampilan (View), dan pengontrol (Controller) dalam sebuah aplikasi. Pendekatan ini membantu dalam memperjelas tanggung jawab masing-masing komponen dan memisahkan perhatian terkait tampilan dengan logika bisnis, sehingga meningkatkan struktur dan modularitas aplikasi.

Dalam konteks penghitungan rata-rata nilai mahasiswa, penerapan MVC dapat memberikan manfaat signifikan. Model akan bertanggung jawab untuk mengelola data nilai mahasiswa, termasuk operasi-operasi perhitungan seperti menjumlahkan nilai dan menghitung rata-ratanya. View, di sisi lain, akan memberikan antarmuka yang interaktif dan informatif kepada pengguna, sehingga mereka dapat melihat rata-rata nilai dengan jelas dan mudah dipahami. Controller akan berperan sebagai penghubung antara Model dan View, menerima input pengguna, dan mengkoordinasikan operasi perhitungan yang diperlukan.

Dalam artikel ini, kami akan menjelaskan secara rinci tentang penerapan Model View Controller (MVC) dalam penghitungan rata-rata nilai mahasiswa. Kami akan membahas langkah-langkah implementasi, struktur komponen, dan hubungan antara Model, View, dan Controller dalam konteks aplikasi ini. Kami juga akan membahas manfaat dari penggunaan pendekatan ini, termasuk

peningkatan struktur kode, pemisahan tanggung jawab, dan kemudahan pengembangan aplikasi di masa mendatang.

Dengan adanya artikel ini, diharapkan pembaca akan memperoleh pemahaman yang lebih baik tentang penerapan MVC dalam penghitungan rata-rata nilai mahasiswa. Kami berharap artikel ini dapat menjadi sumber informasi yang bermanfaat bagi pengembang aplikasi dan praktisi pendidikan yang tertarik untuk meningkatkan efisiensi dan efektivitas dalam pengelolaan data dan perhitungan nilai mahasiswa.

Metode

Dalam aplikasi yang kami buat ini banyak tahapan yang kami lakukan antara lain :

1. Identifikasi Kebutuhan: Langkah pertama dalam penerapan Model View Controller (MVC) adalah mengidentifikasi kebutuhan fungsional dan non-fungsional aplikasi penghitungan rata-rata nilai mahasiswa. Hal ini melibatkan pemahaman yang mendalam tentang persyaratan bisnis, tampilan yang diharapkan, dan operasi perhitungan yang harus dilakukan.
2. Desain Model: Setelah kebutuhan diidentifikasi, langkah berikutnya adalah merancang Model. Model akan bertanggung jawab untuk menyimpan dan mengelola data nilai mahasiswa. Desain Model harus mempertimbangkan struktur data yang tepat untuk menyimpan informasi nilai mahasiswa, serta fungsi-fungsi perhitungan yang diperlukan, seperti penjumlahan nilai dan penghitungan rata-rata.
3. Desain View: Komponen View bertanggung jawab untuk menampilkan informasi kepada pengguna. Dalam konteks penghitungan rata-rata nilai mahasiswa, View harus dirancang untuk menampilkan data nilai mahasiswa secara terstruktur dan jelas. Hal ini dapat meliputi penggunaan tabel, grafik, atau elemen tampilan lainnya yang relevan untuk menampilkan rata-rata nilai.
4. Desain Controller: Controller berperan sebagai penghubung antara Model dan View. Desain Controller harus mempertimbangkan bagaimana menerima input pengguna, seperti tombol atau formulir, dan bagaimana mengoordinasikan operasi perhitungan dengan Model yang sesuai. Controller juga harus memastikan bahwa hasil perhitungan yang dihasilkan oleh Model diteruskan ke View untuk ditampilkan kepada pengguna.
5. Implementasi: Setelah desain Model, View, dan Controller selesai, langkah selanjutnya adalah mengimplementasikannya menggunakan bahasa pemrograman dan framework yang sesuai. Pemilihan bahasa dan framework harus mempertimbangkan kebutuhan dan preferensi pengembang, serta ketersediaan alat yang mendukung penerapan MVC.
6. Integrasi dan Pengujian: Setelah implementasi, komponen-komponen Model, View, dan Controller harus diintegrasikan dalam aplikasi yang utuh. Pengujian perlu dilakukan untuk memastikan bahwa aplikasi berfungsi sesuai dengan kebutuhan dan memberikan hasil perhitungan rata-rata nilai yang akurat.
7. Evaluasi dan Peningkatan: Setelah aplikasi diuji, evaluasi perlu dilakukan untuk mengevaluasi kinerja dan kehandalan aplikasi. Jika ditemukan masalah atau kekurangan, perbaikan dan peningkatan perlu dilakukan dengan memperhatikan feedback dari pengguna atau pemangku kepentingan.
8. Dokumentasi: Langkah terakhir adalah menyusun dokumentasi yang lengkap, termasuk dokumentasi teknis dan petunjuk pengguna, untuk memudahkan penggunaan dan pemeliharaan aplikasi.

Dengan mengikuti langkah-langkah di atas, penerapan Model View Controller (MVC) dalam penghitungan rata-rata nilai mahasiswa dapat dilakukan secara sistematis dan terstruktur. Metode ini membantu memisahkan tanggung jawab, meningkatkan fleksibilitas dan modularitas aplikasi, serta meningkatkan efisiensi dalam pengelolaan data dan perhitungan nilai mahasiswa.

Hasil dan Pembahasan

Gambar 1

```java
        mhs.setRata((mhs.getUh() + mhs.getUts() + mhs.getUas())/3 );

        if(mhs.getRata() >=75){
            mhs.setKompetensi("KOMPETEN");
        }else{
            mhs.setKompetensi("TIDAK KOMPETEN");
        }

        tampil.getTxthasil().setText("========== INFORMASI MAHASISWA ==========\n
                                +"NIM\t: "+mhs.getNim()+"\n"
                                +"NAMA\t: "+mhs.getNama()+"\n"
                                +"UH\t: "+mhs.getUh()+"\n"
                                +"UTS\t: "+mhs.getUts()+"\n"
                                +"UAS\t: "+mhs.getUas()+"\n"
                                +"RATA-RATA\t: "+mhs.getRata()+"\n"
                                +"ANDA DINYATAKAN "+mhs.getKompetensi()+"\n"
                                +"======================================")

    }


    public void reset(){
        tampil.getTxtnim().setText("");
        tampil.getTxtnama().setText("");
        tampil.getTxtuh().setText("0");
        tampil.getTxtuts().setText("0");
        tampil.getTxtuas().setText("0");
        tampil.getTxthasil().setText("");
    }
}
```

Gambar 2

Gambar 1 & 2 : package Controller dengan nama class CtrlMahasiswa

```java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package Model;

/**
 *
 * @author firgo
 */
public class Mahasiswa {
    private String nim;
    private String nama;
    private int uh;
    private int uts;
    private int uas;
    private float rata;
    private String kompetensi;


    public String getNim() {
        return nim;
    }

    public String getNama() {
```

```java
    public int getUts() {
        return uts;
    }

    public int getUas() {
        return uas;
    }

    public float getRata() {
        return rata;
    }

    public String getKompetensi() {
        return kompetensi;
    }

    public void setNim(String nim) {
        this.nim = nim;
    }

    public void setNama(String nama) {
        this.nama = nama;
    }

    public void setUh(int uh) {
        this.uh = uh;
    }

    public void setUts(int uts) {
        this.uts = uts;
    }
```

Gambar 4

Gambar 5

Gambar 3, 4 dan 5 : Package Model dengan nama class Mahasiswa

Gambar 6 : Desain tampilan aplikasi nilai mahasiswa

Gambar 7

```java
        return txtnama;
    }
    public JTextField getTxtuh(){
        return txtuh;
    }
    public JTextField getTxtuts(){
        return txtuts;
    }
    public JTextField getTxtuas(){
        return txtuas;
    }
    public JTextArea getTxthasil(){
        return txthasil;
    }
    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    Generated Code

    private void txtnamaActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    private void txtuhActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    private void txtutsActionPerformed(java.awt.event.ActionEvent evt) {
```

Gambar 8

```java

    private void txtuasActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    private void btnprosesActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        ctrl.proses();
    }

    private void btnresetActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        ctrl.reset();
    }

    private void btnexitActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
       System.exit(0);
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
         Look and feel setting code (optional)

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new FormMahasiswa().setVisible(true);
```

Gambar 9

```java

    // Variables declaration - do not modify
    private javax.swing.JButton btnexit;
    private javax.swing.JButton btnproses;
    private javax.swing.JButton btnreset;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JLabel jLabel6;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JTextArea txthasil;
    private javax.swing.JTextField txtnama;
    private javax.swing.JTextField txtnim;
    private javax.swing.JTextField txtuas;
    private javax.swing.JTextField txtuh;
    private javax.swing.JTextField txtuts;
    // End of variables declaration
}
```

Gambar 10

Gambar 7, 8, 9, dan 10 : Package view dengan nama class FormMahasiswa untuk memberikan perintah pada desain yang sudah dibuat.

Gambar 11 : tampilan aplikasi

Gambar 12 : tampilan aplikasi setelah diisi dan diproses

Kesimpulan

Dalam artikel ini Penerapan Model View Controller (MVC) dalam penghitungan rata-rata nilai mahasiswa merupakan pendekatan yang berguna dan efektif. Dengan memisahkan tanggung jawab antara Model, View, dan Controller, aplikasi dapat dikembangkan dengan lebih terstruktur, modular, dan mudah dikelola. Pendekatan ini memberikan manfaat yang signifikan bagi pengembang dan pengguna aplikasi, meningkatkan efisiensi dalam pengelolaan data dan perhitungan nilai mahasiswa.

Daftar Pustaka

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1995). Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley Professional.

Fowler, M. (2004). Patterns of Enterprise Application Architecture. Addison-Wesley Professional.

Reenskaug, T. (1979). Models—Views—Controllers. Journal of Object-Oriented Programming, 1(3), 26-49.

Burbeck, S. (1992). Applications Programming in Smalltalk-80(TM): How to use Model-View-Controller (MVC). The Journal of Object-Oriented Programming, 5(4), 26-49.

Apple Inc. (2021). Model-View-Controller. Apple Developer Documentation. Retrieved from: https://developer.apple.com/library/archive/documentation/General/Conceptual/DevPedia-CocoaCore/MVC.html

Oracle Corporation. (2021). Model-View-Controller (MVC) Design Pattern. Oracle Java Documentation. Retrieved from: https://docs.oracle.com/javase/8/docs/technotes/guides/gls/design-patterns/model-view-controller.html

Freeman, E., Robson, E., Bates, B., & Sierra, K. (2004). Head First Design Patterns. O'Reilly Media.

Johnson, R. (2008). GUI Architectures. Retrieved from: https://martinfowler.com/eaaDev/uiArchs.html